EXPRESS MAIL LABEL NO:
EV004868661US

# VIDEO STABILIZER

Wei Xiong

## COPYRIGHT NOTICE

A portion of the disclosure of this patent document contains material that is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosure, as it appears in the Patent and Trademark Office patent files or records, but otherwise reserves all copyright rights whatsoever.

## BACKGROUND

### 1. Field of the Invention

The invention is generally related to digital image processing, and, in particular, is related to video stabilization.

### 2. Description of Related Art.

Video cameras are becoming more popular today as they become more widely available at lower prices. A video camera records sequential images within "frames." A frame is a representation of an image at an instant of time. Each frame in a video segment (i.e., a sequence of frames) represents the image at a different instant in time. When several frames are recorded, at sequential instances in time, and are shown to the human eye in quick succession, the human eye is able to discern motion in the video segment.

Video (i.e., moving pictures) normally consist of a lot of motion, including object motion, such as a bird flying, and camera motion, such as camera panning,

PTO/SB/21 (09-04)
Approved for use through 07/31/2006. OMB 0651-0031

zooming, and tilting. The human eye may feel uncomfortable if the motion in the video is too fast, especially if the motion is changing very quickly in its direction or speed.

Other than natural motion of an object (e.g., one that is moving away from a stable or non-moving video camera), the video camera may record unwanted motion. The term "unwanted" usually refers to camera motion frequently resulting from hand jittering or from the camera person walking or otherwise moving. Other unwanted motion may result if, for example, wind shakes a mounted camera or a vehicle carrying a camera traverses rough terrain. Unwanted motion is very common in home videos.

Unwanted motion is also said to include "high frequency" components. The term "high frequency" refers to the motion changing very quickly in direction (e.g., left to right) or in speed (e.g., getting very fast or slowing down) in a unit of time (e.g., one second).

Video stabilization refers to a process for compensating for "unwanted" motion in a video segment. Video stabilization has been discussed in the literature for a number of years. The problem of "shaking shot" is common in home video. When a hand held camera is unsteady, video frames may be blurred or displaced, making viewing of the video difficult. Some high-end hand held video cameras support hardware video stabilization devices, notably "liquid lens" and electronic solutions. Some video cameras with such electronic solutions include the Sony® DCR-TRV900 video camera available from Sony Electronics, Inc. and the Canon® ZR20 video camera available from Canon, U.S.A. Inc.

Such devices incorporating electronic solutions are expensive and are typically unavailable to the casual photographer. Moreover, these electronic solutions handle small amounts of movement. The electronic solutions are unable to distinguish between scanning and a lot of shaking, so they are unable to eliminate shaking. Overall, the electronic solutions try to slow down all motion in the video, without distinguishing between wanted and unwanted motion.

Additionally, video stabilization has been a topic in academic research. In some cases, the motion between each pair of consecutive frames or between each frame and a chosen origin frame is estimated. Based on the motion estimation, some inter-frame transformation parameters are computed for transforming (sometimes referred to as warping) each frame with respect to the origin frame. This technique, however, does not work when the camera is moving normally (e.g., panning, zooming, or tilting). Instead, the technique works only with a static camera.

Another conventional solution for unwanted motion is to map the estimated motion to some camera motion model (e.g., panning, zooming, tilting, etc.). Based on the camera model, each frame is transformed (i.e., its pixels are adjusted). There are several difficulties in these techniques. For example, motion estimation in video is time consuming and inaccurate when there are many outliers (i.e., data points that do not fall in the range of other data points). Another difficulty is that it is not uncommon to have several camera motions simultaneously, and, therefore, application of a single camera motion model leads to inaccurate results.

SUMMARY

A video segment is processed to remove unwanted motion, resulting in a stabilized video segment.

According to one embodiment of the invention, a method for stabilizing motion in a sequence of frames is provided. One or more features in a first frame in the sequence of frames are identified. Tracked positions are calculated for one or more features in each other frame in the sequence of frames based on the features in the first frame. Ideal positions are calculated for the features in each other frame in the sequence of frames based on the tracked positions. Transformation information is identified based on the tracked positions and the ideal positions. Each other frame in the sequence of frames is transformed by adjusting pixels based on the transformation information.

According to another embodiment of the invention, a method for stabilizing a sequence of frames is provided. A first position of a point of interest in a first frame is calculated. Estimated positions of points of interest are identified in a second frame and a third frame that correspond to the point of interest in the first frame. Tracked positions of points of interest are identified in the second frame and the third frame based on the estimated positions of the point of interest. The tracked positions comprise a second position for the point of interest in the second frame and a third position for the point of interest in the third frame. The first position, the second position, and the third position are plotted on an X,Y coordinate graph. The first position is connected to the third position on the X,Y coordinate graph. The ideal positions of the point of interest in the first frame, second frame, and third frame lie on the connection.

EXPRESS MAIL

According to a further embodiment of the invention, a system comprises a computer including a processor and a memory, a sequence of frames stored in the memory, and a program stored in the memory of the computer. The program is executed by the processor of the computer to identify one or more features in a first frame in the sequence of frames and to calculate tracked positions for one or more features in each other frame in the sequence of frames based on the features in the first frame. Execution of the program calculates ideal positions for the features in each other frame in the sequence of frames based on the tracked positions and identifies transformation information based on the tracked positions and the calculated positions. Furthermore, execution of the program transforms each other frame in the sequence of frames by adjusting pixels in each other frame based on the transformation information.

According to yet another embodiment of the invention, a system comprises a camera, a sequence of frames captured by the camera, a computer with a processor and a memory. The sequence of frames is stored in the memory of the computer. The system includes means for performing local tracking to obtain tracked positions for a feature in multiple frames of the sequence of frames and means for calculating ideal positions for the feature in each of the multiple frames. The system also includes means for identifying transformation information based on the tracked positions and the ideal positions for each feature in the one or more features. Additionally, the system includes means for transforming each other frame by adjusting pixels in each other frame based on the transformation information.

PATENT 6230.P037001

According to a further embodiment of the invention, a computer readable storage medium encoded with software instructions. Execution of the instructions performs the following: one or more features in a first frame in the sequence of frames are identified; tracked positions are calculated for one or more features in each other frame in the sequence of frames based on the features in the first frame; ideal positions are calculated for the features in each other frame in the sequence of frames based on the tracked positions; transformation information is identified based on the tracked positions and the calculated positions; and, each other frame in the sequence of frames is transformed by adjusting pixels in each other frame based on the transformation information.

The invention is better understood upon consideration of the detailed description below, and the accompanying drawings.

## BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a block diagram that illustrates components that may be used in one embodiment of the invention;

Figure 2 is a flow diagram that illustrates processing of a video segment in one embodiment of the invention;

Figures 3A, 3B, and 3C illustrate frames in a video segment in one embodiment of the invention;

Figures 4A and 4B are flow diagrams that illustrate video stabilization processing in one embodiment of the invention;

Figure 5 illustrates motion of points of interest of an image when motion is slow versus when motion is fast in one embodiment of the invention;

Figure 6 illiterates lines in a frame in one embodiment of the invention;

Figures 7A, 7B, and 7C illustrate tracking in one embodiment of the invention;

Figures 8A and 8B illustrate sample graphs showing motion trajectories in embodiments of the invention.

Use of the same reference symbols in different figures indicates similar or identical items.

1005935.01

## DETAILED DESCRIPTION

In accordance with an embodiment of the invention, a computer programmed with software (referred to herein as "video stabilization system") processes a video segment to remove unwanted motion, resulting in a stabilized video segment. The video stabilization system tracks one or more features through multiple frames, identifies ideal positions for the features, and then generates transformation information (e.g., rotation, scaling, shearing, and/or translation) to transform all pixels in each frame to ideal positions based on the ideal positions of the features.

In particular, the video stabilization system of the invention tracks some features all the way through some video segment, rather than estimating camera motion between two frames. In one embodiment, a "feature" may be, for example, a point, line, region, edge, etc. For example, a feature may be a point of interest in an image that represents a large brightness change in two dimensions.

A video segment includes a sequence of frames. Each frame represents an image (simplistically, this can be viewed as a picture taken with a camera). If a

sequence of frames is taken of an image that is not moving with a video camera that is not moving, each pair of consecutive frames will be almost exact (note there may be some change due to hand jitter and other factors).

On the other hand, if the sequence of frames are taken of a moving object, or the video camera is moving, or both, consecutive frames capture different images. If there is smooth motion, then it is possible to select a feature in a first frame, find the corresponding pixel or pixels associated with that feature in subsequent frames, map the points on a graph (e.g., an X-Y graph), and obtain a smooth path when the positions of the features are connected. When there is unwanted motion, however, a feature across frames does not follow a smooth path. This results in "jitters" when viewed by the human eye. The video stabilization system of the invention identifies the trajectory of a set of features and adjusts them to "ideal positions" that results in a smooth video segment.

The video stabilization system reduces and/or removes the unwanted motion (also referred to as a "high frequency portion") without affecting normal motion (also referred to as a "low frequency portion"). Therefore, a camera need not be ideally static or have only one type of motion.

The software solution has many advantages over a hardware solution (e.g., liquid lens). For example, the video stabilization system can classify the "normal" camera motion and jittering or unwanted motion. The video stabilization system can correct larger displacement (e.g., caused by a fast moving camera) than hardware. The video stabilization system can stabilize video from any video camera, rather than from high-end products, as is the case with hardware solutions. Furthermore, software production also costs less than hardware manufacturing.

Figure 1 is a block diagram that illustrates components that may be used in one embodiment of the invention. In one embodiment, a video camera 100 records an image 110. The video camera 100 and/or the image may be in motion. The video camera 100 includes data storage that stores the video segment. The video segment is transferred to a computer 120, which includes video stabilization system 130 stored, for example, on a hard drive of the computer 120 or a CD-ROM (compact disc-read only memory) inserted into the CD-ROM drive of the computer 120. In other embodiments, the video stabilization system 130 may be implemented as a hardware video stabilizer or as a combination of hardware and software.

In one embodiment, the video camera 100 is a digital video camera. Digital video cameras offer many advantages. For example, digital images are easier to manipulate and easier to distribute over electronic media (e.g., the Internet or e-mail). In another embodiment, the video camera 100 is an analog video camera using film to record images. The film can be converted to digital images for processing with the video stabilization system 130. In yet another embodiment, a still picture camera, rather than video camera 100, is used to take a series of pictures that are either digitally recorded or converted to digital images. The series of pictures are transformed into a video segment that may be processed with the video stabilization system 130.

The video stabilization system works with both gray scale or color images. For example, each image can be a two-dimensional array of RGB (red-green-blue) pixels or YUV pixel values representing color pixels. YUV is defined by the Commission International de L'Eclairage (CIE), which is an international

committee for color standards. YUV is often used in Phase Alternation Line (PAL) television (an analog television display standard), where the luminance and the chrominance are treated as separate components. In YUV systems, a luminance signal (represented with "Y") typically occupies the maximum bandwidth, while chrominance signals (represented by "U" and "V") typically occupy half the bandwidth each (i.e., because the eye is less sensitive to color detail).

In one embodiment, the images are represented in Microsoft Windows™ 24-bit BITMAP format. In this format, each pixel has three adjacent bytes for Blue, Green, and Red channels respectively. In one embodiment, each of the source images is W (i.e., width) by H (i.e., height) pixels. For example, the dimensions may be 720x480 pixels or 352x288 pixels.

Figure 2 is a flow diagram that illustrates processing of a video segment in one embodiment of the invention. In particular, block 200 represents the video stabilization system 130 receiving a video segment. Block 202 represents the video stabilization system 130 performing video stabilization to remove unwanted motion from the video segment. For example, the video stabilization system 130 processes the data to reduce the jitter caused by unintended camera motion. Block 204 represents the video stabilization system 130 outputting a stabilized video segment. The stabilized video segment may be output to local data storage, remote data storage, to a user monitor, to television transmitters, or to another device. The digital images may be converted to analog images. Also, the stabilized video segment may be output to other software for additional processing.

In an alternative embodiment, as a video camera 100 captures images, the video camera 100 transfers data directly to computer 120, which has sufficient memory to hold the data. The computer 120 processes the data in real time to reduce jitter, and, for example, transfers the data to storage, to a user monitor, or to television transmitters.

The computer 120 may be a personal computer, workstation, laptop computer, personal digital assistant, mainframe computer, or other processing device. Also, the computer 120 may be a general purpose or a special purpose computer. For example, computer 120 may be a computer having a Pentium® chip, available from computer vendors such as International Business Machines Corporation, Inc., of Armonk, New York or Apple Computer, Inc. of Cupertino, California. The computer 120 may include an operating system, such as Microsoft ® Windows ® 2000 from Microsoft Corp. of Redmond, Washington.

Given ongoing advances in the data processing industry, it is conceivable that the storage and processing features illustrated in Figure 1 may be incorporated on integrated circuits, microprocessors, and other electronics small enough to fit within a handheld video camera. Therefore, the video stabilization system 130 may be incorporated into the video camera 100 as software, hardware, or a combination of hardware and software. Nevertheless, merely reducing the size or altering the location of elements of this invention does not depart from the spirit and scope of the invention.

Figures 3A, 3B, and 3C illustrate frames in a video segment in one embodiment of the invention. In Figure 3A, a scene is the subject of a video camera recording resulting in frame 300. In the scene, one or more objects (e.g.,

cars) may have been in motion. Additionally, the video camera may have been in motion (e.g., panning, zooming or tilting). This is referred to as normal motion. In addition, there may be unintended motion of the camera caused by, for example, the camera person walking while recording.

In Figure 3A, multiple points of interest (POIs) have been detected in frame 300. Although points of interest are illustrated in Figures 3A, 3B, and 3C, other features may be used in accordance with the technique of the invention. In this illustration, crossbars (i.e., +) are used to indicate points of interest, such as 310, 312, and 314. In one embodiment, at least 3 points of interest are tracked through a video segment. Figures 3B and 3C illustrate subsequent frames to Figure 3A in a video segment.

Figures 4A and 4B are flow diagrams that illustrate video stabilization processing in one embodiment of the invention. For ease of illustration, the flow diagrams refer to a point of interest, which is one type of feature. It is to be understood that the technique of the invention is applicable to any type of feature or distinguishing characteristic between frames.

Block 400 represents the video stabilization system 130 initializing memory and variables. Initialization is performed to initialize variables, allocate memory, and open an input video file, such as an Audio Video Interleaved (AVI) file. In one embodiment, a video file is one in which motion picture and audio are interleaved.

Block 401 represents the video stabilization system 130 determining whether the video camera motion was fast or slow. In one embodiment, a user submits input indicating whether the camera moved fast or slow. In another

embodiment, the video stabilization system 130 compares one or more points in a series of frames and determines whether the motion was fast or slow based on the amount of distance each point moved from one frame to another.

For example, if the video stabilization system 130 finds that point (3,3) in frame 1, is in location (30,30) in frame 2, the video stabilization system 130 detects that the camera motion was fast. While, for example, if the same point (3,3) in frame 1 is in location (4,4) in frame 2, the video stabilization system 130 detects that the camera motion was slow.

Figure 5 illustrates motion of points of interest of an image when motion is slow versus when motion is fast in one embodiment of the invention. Image 500 is a first image having three points of interest 502, 504, and 506. Image 510 represents a second image captured when motion is slow, and points of interest 502, 504, and 506 have moved horizontally relative to their positions in image 500. Image 520 represents a third image captured when motion is fast, and points of interest 502, 504, and 506 have moved horizontally relative to their positions in image 500. In image 520, point of interest 506 has partially moved off of the image. Points of interest 502, 504, and 506 have moved more in image 520 than in image 510.

The video stabilization system 130 processes groups of pictures in the video segment. Block 402 represents the video stabilization system 130 retrieving a sequence of frames of a video segment from storage or memory. In particular, the video stabilization system 130 selects a video segment made up of a sequence of frames and designates this as a group of pictures (GOP). In one embodiment, a group of pictures may include frames captured during 2-3 seconds of recording,

and approximately 25-30 frames may be captured each second. So, a group of pictures may include 50-90 frames.

In another embodiment, a set number of frames (e.g., 60 frames) are selected as a group of pictures. At this point, the length of the group of pictures is calculated.

In either embodiment, the video stabilization system 130 may start with a long video segment (e.g., 3 seconds or 90 frames). If the video stabilization system 130 determines that many (e.g., more than half) of the points of interest in the first frame are missing from the last frame in the sequence, then the video stabilization system 130 may truncate the video segment and process a smaller sequence of frames. Then, the next video segment processed would begin at the truncated position. For example, if the video stabilization system 130 initially took a video segment having 90 frames and truncated the video segment to work with 40 frames, the next video segment would start at the 41st frame.

Block 404 represents the video stabilization system 130 determining whether the video segment has been completely processed (i.e., to its end). If so, processing continues to block 406. Otherwise, processing continues to block 440 and post-processing is performed. Post-processing includes freeing memory and closing the output .avi file.

Block 408 represents the video stabilization system 130 computing points of interest (POIs) for the first frame. For example, Figure 3A illustrates a first frame with points of interest. These points of interest may be selected using, for example, the Harris technique for corner and edge detection or the Kitchen-Rosenfeld technique for corner detection. For more information on the Harris

technique, see "A Combined Corner and Edge Detector," by C. Harris and M. Stephens, Fourth Alvey Vision Conference, pp. 147-151, 1988, which is described in "Feature Point Extraction" at http://www.esat.kuleuven.ac.be/~pollefey/tutorial/node51.html, which is incorporated herein by reference in its entirety and is listed in Appendix A. For more information on the Kitchen-Rosenfeld technique, see "*Gray-Level Corner Detection*," by L. Kitchen and A. Rosenfeld, Pattern Recognition Letters, 95-102, December, 1982, which is incorporated herein by reference in its entirety.

Alternatively the Hough transform technique for line detection (rather than point of interest detection) may be used. For example, if the Hough transform technique were used, lines would be detected and compared in frames, rather than points of interest. Figure 6 illiterates lines in a frame 600 in one embodiment of the invention. For example, vertical line 602 is found along a telephone pole, horizontal line 604 is found along a sidewalk, and horizontal lines 606 and 608 are found along parking space markings. Although the lines depicted in Figure 6 extend across much of frame 600, the lines could be shorter segments in alternative embodiments.

For more information on the Hough transform technique, see "Fundamentals of Digital Image Processing," by Anil K. Jain, Prentice-Hall, Inc., page 362, 1989 or "Digital Image Processing," by Rafael C. Gonzalez and Richard E. Woods, page 432-438, each of which is incorporated herein by reference in its entirety. In one embodiment, the positions of the points of interest of the first frame may be stored in a linked list. However, alternative data structures such as tables may be used. For ease of illustration, a table will be used to provide an

example. The following Table A is a sample table of positions of points of interest in a first frame (e.g., frame 300 of Figure 3A):

Table A

| **Frame 1** | X-coordinate | Y-coordinate |
|---|---|---|
| Point 1 | 7 | 3 |
| Point 2 | 8 | 7 |
| Point 3 | 9 | 8 |

For each pair of consecutive frames, the video stabilization system 130 identifies positions for corresponding points of interest (blocks 410-426). In particular, the video stabilization system 130 identifies estimated positions for points of interest in blocks 418 and 424. Then, in block 420, the video stabilization system 130 identifies tracked positions for the points of interest. The tracked positions are more accurate than the estimated positions.

In an alternative embodiment, a selected frame, such as the first frame or the middle frame in a group of pictures, may be compared to every other frame. This alternative works well when there is little change between the compared frames. Block 410 represents the video stabilization system 130 selecting the next frame in the group of pictures, starting with the second frame. Block 412 represents the video stabilization system 130 setting a current-frame variable to the selected next frame and setting a previous-frame variable to the frame before the current-frame. Initially, the video stabilization system 130 works with a first frame and a second frame (which represent the previous and current frames, respectively, in one iteration of the process).

Block 414 represents the video stabilization system 130 determining whether all the frames in the group of pictures have been selected and processed. If so, processing continues to block 427. Otherwise, processing continues to block 416. Block 416 represents the video stabilization system 130 determining whether the camera was moving fast when the video segment was recorded (this information is determined in block 401).

If the intended motion of the camera (e.g., pan, zoom, or tilt) is slow, then global tracking is performed in the second frame with respect to the points of interest selected for the first frame. If the motion is fast, a new set of points of interest are detected in the second frame and matching (rather than tracking) is performed in the first frame for these points of interest. In one embodiment, global tracking refers to selecting a point of interest in a first frame, such as (3, 3), and searching a large area in the second frame around this point, such as searching a 16X16 area around (3, 3) in the second frame, for a point of interest corresponding to the point of interest in a first frame. In one embodiment, matching refers to selecting points of interest in a first frame, selecting points of interest in a second frame, and attempting to match the points of interest in the first frame to those in the second frame.

In one embodiment, frames are examined in consecutive pairs. One pair includes a first frame and a second frame (which are also referred to as a previous frame and a current frame, respectively).

In block 418, the video stabilization system 130 performs global tracking. There are many techniques that may be used to perform tracking (global or local). One technique is described in "Making Good Features Track Better,

by Tiziano Tommasini, Andrea Fusiello, Emanuele Trucco, and Vito Roberto, pages 1-6, which is incorporated herein by reference in its entirety. For global tracking, the points of interest from the first frame are tracked in a large area in the second frame to estimate the global movement between the first frame and the second frame. For global tracking, initially, a first point of interest in a first frame is selected. For this point of interest, the video stabilization system 130 attempts to find a point of interest within an area of a second (e.g., consecutive) frame that estimates the position of the point of interest in the first frame. This estimated position of the point of interest in the second frame may not be the point of interest that actually corresponds to the point of interest in the first frame due to possible errors with the tracking technique. The estimated position is only an estimate of where the point of interest from the first frame is in the second frame, considering there may have been camera and/or object motion. As will be discussed with respect to local tracking in block 420, additional processing is done to confirm that the estimated position of the point of interest actually corresponds to the point of interest in the first frame.

For example, the video stabilization system 130 may search a 16X16 pixel area around the location of the first point of interest in the second frame. For example, if the first point of interest is (8, 8) in the first frame, then the area formed by corner points (0, 0), (0, 16), (16, 0), and (16, 16) may be searched for a corresponding point of interest in the second frame. The result is an estimated position of the point of interest in the second frame.

Figures 7A, 7B, and 7C illustrate tracking in one embodiment of the invention. Once, point of interest is identified in a first frame, such as frame 700 in

Figure 7A, then a 3X3 "window" is used to obtain the pixel value of each pixel in the 3X3 area 710 in which the point of interest 702 is the center pixel. Each pixel has a value associated with it. For example, when an image represented with a two-dimensional array of RGB (red-green-blue) pixels, each pixel has a value associated with it ranging from 0-255. A larger pixel value may be indicative of, for example, a brighter, more colorful pixel. In alternative embodiments, the area 710 selected may be smaller or larger than 3X3.

Next, for a second frame 720 in Figure 7B, the video stabilization system 130 selects a large area, such as area 721. Within 8X8 area 721, for each pixel, the video stabilization system 130 selects a pixel, designates the selected pixel as a center pixel of a 3X3 area, and identifies the pixel value of each pixel in the 3X3 area. For example, for pixel 722, the video stabilization system identifies the pixel values of each pixel in the 3X3 area designated by brackets 724 and 726. Once, this is done for each pixel in area 721, the video stabilization system attempts to find the pixel in the second frame 720 which is most similar to the point of interest 702 in the first frame 700.

In one embodiment, the video stabilization system 130 will compare the differences between the 3X3 area 710 in the first frame with each 3X3 area surrounding each pixel in the second frame to determine which pixel in the second frame has "neighboring" pixels (i.e., pixels that are one position away from the center pixel in any direction) that are most similar to the point of interest in the first frame.

After global tracking, the video stabilization system 130 has identified estimated positions for the points of interest in the second frame that estimate

points of interest in the first frame. At this time, these are estimated positions for points of interest because the positions could be inaccurate. That is, the tracking technique of looking at neighboring pixel values may result in finding a pixel in the second frame that does not correspond to the point of interest in the first frame. Local tracking, performed in block 420, attempts to confirm whether global tracking found the correct position of the point of interest in the second frame. Local tracking is described below with reference to Figure 7C.

In one embodiment, the estimated positions of points of interest may be stored in a linked list. However, alternative data structures such as tables may be used. For ease of illustration, a table will be used to provide an example. The following Table B is a sample table of estimated positions of points of interest in a second frame that correspond to points of interest in the first frame (illustrated in Table A) after global tracking has been performed (block 418).

Table B - Estimated Positions from Global Tracking

| **Frame 2** | X-coordinate | Y-coordinate |
|---|---|---|
| Point 1 | 8 | 3 |
| Point 2 | 9 | 8 |
| Point 3 | 10 | 9 |

In one embodiment, global tracking may be performed using epipolar geometric constraints to narrow down the tracking area and to remove outliers (i.e., points of interest that are far away from other points of interest). For more information on epipolar geometry, see "A Robust Technique for Matching Two Uncalibrated Images Through the Recovery of the Unknown Epipolar Geometry,"

by Zhengyou Zhang, pages 1-38, May 1994, which is incorporated herein by reference in its entirety.

On the other hand, if camera motion is fast, a new set of points of interest are selected in the second frame (block 422). Then, for each of these points, the video stabilization system 130 attempts to match points of interest in the second frame to points of interest in the first frame to estimate global movement between the first and second frames (block 424).

In one embodiment, for two frames, points of interest are calculated in each frame. Then, for each point of interest in each frame, the video stabilization system 130 computes a neighbor (i.e., another point of interest in that frame) distribution in terms of distance and direction. For example, in Figure 3C, for frame 350, the video stabilization system 130 identifies an area 360 around point of interest 362. The area 360 may be, for example, a circular area having a specified radius. Then, the video stabilization system 130 generates a distance histogram (indicating the distance of each neighbor in the area 360 from the point of interest) and a direction histogram (indicating the direction of each neighbor, such as above and to the left, relative to the point of interest). The area 360 may be partitioned into sections for ease in generating the histograms. The histograms are generated by the video stabilization system 130 for each point of interest in the first and second frames. The video stabilization system 130 determines that a point of interest in a first frame corresponds to a point of interest in a second frame if they have the most similar neighbor distributions (e.g., two points of interest have neighbors at approximately the same distance and direction away from them). In

this example, point of interest 362 at estimated position (7, 4) corresponds to point of interest 310 in the first frame illustrated in Figure 3A.

In an alternative embodiment, the matching technique described in "A Fast Matching Method for Color Uncalibrated Images Using Differential Invariants," British Machine Vision Conference, by V. Gouet, P. Montesinos, D. Pel, pages 367-376 may be used, and this article is incorporated herein by reference in its entirety.

In other embodiments, matching can be implemented using any correlation technique known in the art. For example, a description of image matching by correlation can be found in "Digital Image Processing" by R. Gonzales and R. Woods, Addison-Wesley Publishing Co., Inc., 1992, pages 583-586. Correlation is a technique for matching two images by finding, for an observed region on the first image, the corresponding region on the second image. Correlation may be performed by selecting an observed region defined by a window in the first image. Then, the second image is searched by moving the window around the entire image area and computing the correlation between the observed region and each of the areas in the second image. The similarity between two areas are determined using correlation criteria and a match between the two areas is found when the correlation yields the largest correlation value. When a match between the observed region and the match region is found, the distance from the observed point and its match is called "disparity" having units in pixels.

In another embodiment, the correlation process further includes a brightness normalization operation to account for the fact that first and second images may be captured at different times with different exposure parameters.

Block 426 represents the video stabilization system 130 updating the current positions of points of interest to their estimated positions in a data structure. In one embodiment, this information may be stored in a linked list. However, alternative data structures such as tables may be used. For ease of illustration, a table will be used to provide an example. In this example, The following Table C is a sample table of estimated positions of points of interest in a second frame that match points of interest in the first frame (illustrated in Table A):

Table C - Estimated Positions from Matching

| **Frame 2** | X-coordinate | Y-coordinate |
|---|---|---|
| Point 1 | 7 | 4 |
| Point 2 | 9 | 7 |
| Point 3 | 10 | 8 |

Whether camera motion is slow or fast, it is possible that a first point of interest in a first frame is no longer in the second frame (i.e., it has moved completely off the frame). Therefore, it may be that not every point of interest in one frame has a corresponding point of interest in a second frame. Additionally, there may be new points of interest in the second frame that do not have corresponding points of interest in the first frame.

Next, the video stabilization system performs local tracking (block 420) on estimated positions of points of interest. In one embodiment, local tracking may be performed using epipolar geometric constraints to narrow down the tracking area and to remove outliers (i.e., points of interest that are far away from other points of

interest). For more information on epipolar geometry, see "A Robust Technique for Matching Two Uncalibrated Images Through the Recovery of the Unknown Epipolar Geometry," by Zhengyou Zhang, pages 1-38, May 1994, which is incorporated herein by reference in its entirety.

In particular, the estimated positions of points of interest (from either global tracking in block 418 or matching in block 424) are tracked again, but in a smaller area, such as an area 2X2 pixels in size. For example, in frame 730 of Figure 7C, 2X2 area 731 is tracked. That is, the 3X3 window used for frame 700 is applied to each pixel in the 2X2 area 731. Each pixel in area 731 is treated as the center of a 3X3 area, and the values of the pixels in the 3X3 area are identified. Then, the pixel values in each 3X3 area are compared to the pixel values in the 3X3 area 710 of frame 700. The result may be that the estimated position of a point of interest in a second frame is found to be the correct position that corresponds to the point of interest in the first frame, taking into consideration motion of the camera and/or object. Thus, local tracking adjusts the estimated position of points of interest for improved accuracy, resulting in tracked positions of points of interest.

The result of blocks 410-426 is a table illustrating, for each point of interest, a tracked position for each corresponding point of interest in each subsequent frame.

The following Table D is a sample table illustrating tracked positions of points of interest for a second frame after local tracking has been performed (block 420):

Table D - Tracked Positions from Local Tracking

| **Frame 2** | X-coordinate | Y-coordinate |
|---|---|---|
| Point 1 | 9 | 4 |
| Point 2 | 11 | 8 |
| Point 3 | 10 | 8 |

After determining the motion of the points of interest, the video stabilization system 130 draws a motion trajectory (block 427) for each point of interest across multiple frames. In particular, the video stabilization system 130 selects a point of interest. For the selected point of interest, the video stabilization system 130 plots the tracked positions (identified in block 420) of the point of interest across a sequence of frames on an X,Y coordinate graph. Next, the video stabilization system 130 connects the first position to the last position in the graph using a linear or non-linear connection (e.g., a line or a curve, respectively). The linear or non-linear connection is referred to as a motion trajectory. All positions on the motion trajectory are considered to be ideal positions, while positions of the point of interest that do not fall on the motion trajectory are considered to include unwanted motion.

Figures 8A and 8B illustrate sample graphs 800 and 850 showing motion trajectories in embodiments of the invention. Each graph 800 and 850 illustrates a motion trajectory for one point of interest across five frames. The motion

trajectories are drawn using the tracked positions of one point of interest. If each of the positions were connected together, the result may be a line that is not smooth, which represents a combination of normal motion and unwanted motion. The video stabilization system 130 removes the unwanted motion from the combination, leaving only the normal motion. The normal motion of a point of interest may be linear (e.g., Figure 8A) or nonlinear (e.g., Figure 8B).

In one embodiment, as illustrated by Figure 8A, the video stabilization system 130 uses linear interpolation for each point of interest across five frames in a group of pictures. In particular, after plotting the tracked positions of one point of interest, the video stabilization system 130 draws a line from the first position of the point of interest to the last position of the point of interest. Then, the video stabilization system 130 determines that the ideal positions for a point of interest across the frames fall on the line.

In graph 800, a motion trajectory for one point of interest, referred to as Point 1, is illustrated. For this example, Point 1 has position (7, 3) in a first frame (Table A), tracked position (11, 8) in a second frame (Table D), tracked position (19, 8) in a third frame, tracked position (27, 6) in a fourth frame, and tracked position (35, 13) in a fifth frame. These positions are plotted in graph 800. Then, a line is drawn that connects the position of Point 1 in the first frame of the video segment to the position of Point 1 in the fifth frame of the video segment. In graph 800, the combined motion is represented by line 810, while the normal motion is represented by line 820. For the tracked position (11, 8) 830 of the point of interest in the second frame (labeled "Frame 2"), the ideal position is (11, 5) 840. The difference between the tracked position (11, 8) and the ideal position (11, 5),

which is a vertical 3 pixel change, represents the unwanted motion that is removed. In one embodiment, the ideal position of a point of interest has the same X-coordinate and a different Y-coordinate when compared to the tracked position of the same point of interest.

Table E is a sample table illustrating ideal positions of one point of interest across three frames:

Table E - Ideal Positions

| **Point 1** | X-coordinate | Y-coordinate |
|---|---|---|
| Frame 1 | 7 | 3 |
| Frame 2 | 11 | 5 |
| Frame 3 | 19 | 7 |
| Frame 4 | 27 | 10 |
| Frame 5 | 35 | 13 |

In an alternative embodiment, a curve may be drawn to connect the positions of the points of interest with a smoothening technique. Figure 8B illustrates that the normal motion of a point of interest may be nonlinear. For example, in graph 850, the combined motion is represented by line 860, while the normal motion is represented by curve 870. The tracked position 880 of the point of interest in the second frame (labeled "Frame 2") differs from the ideal position of the point of interest 890.

Once the motion trajectories are drawn (block 427), each frame in a group of pictures is again examined. Block 428 represents the video stabilization system 130 selecting the next frame in the current group of pictures, starting with the first

frame. Block 432 represents the video stabilization system 130 computing ideal positions of each point in a frame according to motion trajectories.

Image stabilization is completed by computing transform parameters from motion trajectories (block 434) for each frame and by transforming the current frame by moving points from current positions to ideal positions (block 436) using the transform parameters. Then, the transformed frame is output (block 438).

The transform parameters are calculated using affine transformation. For more information on affine transformation, see "Computer Graphics Principles and Practice," by James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes, Second Edition, Addison-Wesley Publishing Company, page 207, 1990; "Affine Transformation," at http://www.dai.ed.ac.uk/HIPR2/affine.htm, pages 1-8, printed October 25, 2001 and listed in Appendix B; and "Affine Transform Matrices," http://www.gnome.org/~mathieu/libart/libart-affine-transformation-matrices.htm, pages 1-7, printed October 25, 2001 and listed in Appendix C; each of which is incorporated herein by reference in its entirety.

A matrix is used to solve the transform parameters "a," "b," "c," "d," "m," and "n" for the ideal positions of points of interest (x', y') and the tracked positions of points of interest (x, y). The ideal positions of points of interest (x', y'), are determined by the video stabilization system 130 in block 432 (e.g., see Table E). The tracked positions of points of interest (x, y) are determined by the video stabilization system 130 in block 420 (e.g., see Table D). An image being recorded may be represented using an X, Y coordinate graph. The transform parameters "a," "b," "c," and "d" represent rotation, scaling, and/or shearing information, while the transform parameters "m" and "n" represent translation

information. Once the transform parameter values are obtained for the points of interest, these values are applied to all pixels in the frame. The result is a frame in which unwanted motion has been removed.

A general affine transformation may be written as Equation (1):

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} m \\ n \end{pmatrix} \quad (\_1\_)$$

For example, in order to transform a tracked position of a point of interest such as (11, 8) to an ideal position of the point of interest (11, 5), the transform parameters are calculated using equation (2).

$$\begin{pmatrix} 11' \\ 5' \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} 11 \\ 8 \end{pmatrix} + \begin{pmatrix} m \\ n \end{pmatrix} \quad (\_2\_)$$

Equation (1) may be written with pairs of equations (3) for three pairs of points of interest. Each pair of points of interest includes a tracked position of the point of interest (e.g., $(x_1, y_1)$) and an ideal position of the point of interest (e.g., $(x'_1, y'_1)$).

$$\begin{aligned} x'_1 &= ax_1 + by_1 + m \\ y'_1 &= cx_1 + dy_1 + n \\ x'_2 &= ax_2 + by_2 + m \\ y'_2 &= cx_2 + dy_2 + n \\ x'_3 &= ax_3 + by_3 + m \\ y'_3 &= cx_3 + dy_3 + n \end{aligned} \quad (\_3\_)$$

For example, for tracked position of a point of interest such as (11, 8) and ideal position of the point of interest (11, 5), equations (4) may be used to solve for the transform parameters:

$$11 = 11a + 8b + m \qquad (4)$$
$$5 = 11c + 8d + n$$

The pairs of equations (3) may be rewritten as equation (5):

$$\underbrace{\begin{pmatrix} x_1 & y_1 & 0 & 0 & 1 & 0 \\ 0 & 0 & x_1 & y_1 & 0 & 1 \\ x_2 & y_2 & 0 & 0 & 1 & 0 \\ 0 & 0 & x_2 & y_2 & 0 & 1 \\ x_3 & y_3 & 0 & 0 & 1 & 0 \\ 0 & 0 & x_3 & y_3 & 0 & 1 \\ x_4 & y_4 & 0 & 0 & 1 & 0 \\ 0 & 0 & x_4 & y_4 & 0 & 1 \\ x_5 & y_5 & 0 & 0 & 1 & 0 \\ 0 & 0 & x_5 & y_5 & 0 & 1 \\ \vdots & & & & & \end{pmatrix}}_{M} \underbrace{\begin{pmatrix} a \\ b \\ c \\ d \\ m \\ n \end{pmatrix}}_{a} = \underbrace{\begin{pmatrix} x_1' \\ y_1' \\ x_2' \\ y_2' \\ x_3' \\ y_3' \\ x_4' \\ y_4' \\ x_5' \\ y_5' \\ \vdots \end{pmatrix}}_{b} \qquad (5)$$

Linear equation (5) may be written as equation (6) as follows:

$$Ma = b \qquad (6)$$

The matrix M contains x rows and y columns. If the number of rows exceeds the number of columns, then a linear equation is referred to as over-constrained. For example, for more than three points of interest, there is an over-constrained linear system illustrated in equation (5), which is rewritten as equation (6). The solution to over-constrained equation (5) may not exist in an algebraic sense, but it is valuable to determine a solution in an approximate sense. The error in this approximate solution is then illustrated with equation (7):

$$E = Ma - b \qquad (7)$$

In one embodiment, the approximate solution is selected by optimizing this error in some manner. One useful technique for optimizing this error is referred to as the least squares method. For more information about the least squares method, see, for example, "The Method of Least Squares," engineering fundamentals, at http://www.efunda.com/math/leastsquares/leastsquares.cfm, printed October 25, 2001 and listed in Appendix D, which is incorporated herein by reference in its entirety. The least squares method minimizes the residual sum of squares (rss), which is represented as equations (8)-(10) follows:

$$rss = e^T e \quad (8)$$

$$= [a^T M^T b^T][Mab] \quad (9)$$

$$= a^T M^T Ma\ 2\ a^T M^T b + b^T b \quad (10)$$

Setting the partial derivative of rss with respect to a (i.e., $\partial rss/\partial a$) to zero gives equation (11):

$$\partial rss/\partial a = 2M^T Ma - 2M^T b = 0 \quad (11)$$

Thus, solving for $M^T Ma = M^T b$ or $a = (M^T M)^{-1} M^T b$, the solution vector "a" provides the solution of Ma = b according to the least squares method. Note that $M^{-1}$ represents inversion of matrix M, while $M^T b$ represents transposition.

For example, for tracked position of a point of interest such as (11, 8) and ideal position of the point of interest (11, 5), the video stabilization system 130 may apply the least squares method to determine that the transform parameter values are: a=1, b=0, c=0, d=1, m=0, and n=-3.

For each frame all image elements are then transformed to their ideal positions (block 436). In particular, the values of "a," "b," "c," "d," "m," and "n," which provide rotation, scaling, shearing, and/or translation information (referred

to as transformation information), are applied to each pixel of a frame. The result is a frame in which unwanted motion has been removed. Equation (12) may then be used for each pixel in the first frame, represented as (xf, yf), to obtain its position in the second frame, represented as (xs, ys). In one embodiment, the first frame and the second frame are consecutive frames.

$$\begin{pmatrix} xs \\ ys \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} xf \\ yf \end{pmatrix} + \begin{pmatrix} 0 \\ -3 \end{pmatrix} \tag{12}$$

Continuing with Figure 4B, block 440 represents the video stabilization system 130 determining whether all frames in the group of pictures have been selected. If so, processing continues to block 402. Otherwise, the next frame is selected and processing continues to block 428.

In an alternative embodiment, frame averaging and/or frame sharpening may be performed after block 402 and/or after block 436 to remove noise and detect more interesting points of interest. In particular, some details in a video segment may be unclear due to noise, motion blur, and other distortions. Frame averaging may be used to combine data across frames when a scene is relatively stationary to about one pixel or less of variation across several frames at a time. After frame averaging, some degree of blur may remain (e.g., the stabilized frames are imperfectly aligned with one another). Frame sharpening refers to using a sharpen filter to remove this blur. Some additional fine tuning to adjust brightness offset may also be used in yet other alternative embodiments.

Noise in an image usually refers to the pixels whose values are not related to (i.e., not part of) ideal image content. For example, a recorded image may show

reflection of sunlight, which is "noise" that may be removed. Since most of the image noise is related to single points, frame averaging can reduce the effect of noise, while frame sharpening can increase image contrast. The result of frame averaging and frame sharpening provides an image in which it is easier to detect real feature points. In particular, if an image has a lot of noise, some techniques for finding feature points may select some points that are not actually features.

Microsoft and Windows 2000 are trademarks of Microsoft, Inc. of Redmond, WA.

Although the invention has been described with reference to particular embodiments, the description is only an example of the invention's application and should not be taken as a limitation.

Additionally, the invention may be tangibly embodied as software in a computer-readable device or media, such as memory, data storage devices, and/or data communication devices, thereby making a product or article of manufacture according to the invention. As such, the terms "article of manufacture" and "computer program product" and "computer-readable storage medium" as used herein are intended to encompass software accessible from any computer readable device or media. Using the present specification, the invention may be implemented as a machine, process, or article of manufacture by using programming and/or engineering techniques to produce software, firmware, hardware, or any combination thereof.

Various other adaptations and combinations of features of the embodiments disclosed are within the scope of the invention as defined by the claims.

# APPENDIX A

EP0983404

# Feature point extraction

There are two important requirements for feature points. First, points corresponding to the same scene points should be extracted consistently over the different views. If this were not the case, it would be impossible to find correspondences amongst them. Secondly, there should be enough information in the neighborhood of the points so that corresponding points can be automatically matched. Many feature point extractors have been proposed [120,59,52,31,180].

In our system the *Harris corner detector* [59] is used. Consider the following matrix

$$\mathbf{M} = \begin{bmatrix} \left(\frac{\partial I}{\partial x}\right)^2 & \left(\frac{\partial I}{\partial x}\right)\left(\frac{\partial I}{\partial y}\right) \\ \left(\frac{\partial I}{\partial x}\right)\left(\frac{\partial I}{\partial y}\right) & \left(\frac{\partial I}{\partial y}\right)^2 \end{bmatrix} \tag{D1}$$

where $I(x,y)$ is the grey level intensity. If at a certain point the two eigenvalues of the matrix $\mathbf{M}$ are large, then a small motion in any direction will cause an important change of grey level. This indicates that the point is a corner. The corner response function is given by:

$$R = \det\mathbf{M} - k(\mathrm{trace}\mathbf{M})^2 \tag{D2}$$

where $k$ is a parameter set to 0.04 (a suggestion of Harris). Corners are defined as local maxima of the cornerness function. Sub-pixel precision is achieved through a quadratic approximation of the neighborhood of the local maxima.

To avoid corners due to image noise, it can be interesting to smooth the images with a Gaussian filter. This should however not be done on the input images, but on images containing the squared image derivatives (i.e. $\left(\frac{\partial I}{\partial x}\right)^2$, $\left(\frac{\partial I}{\partial y}\right)^2$, $\left(\frac{\partial I}{\partial x}\right)\left(\frac{\partial I}{\partial y}\right)$).

In practice often far too much corners are extracted. In this case it is often interesting to first restrict the numbers of corners before trying to match them. One possibility consists of only selecting the corners with a value $R$ above a certain threshold. This threshold can be tuned to yield the desired number of features. Since for some scenes most of the strongest corners are located in the same area, it can be interesting to refine this scheme further to ensure that in every part of the image a sufficient number of corners are found.

In figure 4.1 two images are shown with the extracted corners. Note that it is not possible to find the corresponding corner for each corner, but that for many of them it is.

**Figure 4.1:** Two images with extracted corners

Next | Up | Previous | Contents

**Next:** similarity measures **Up:** Relating images **Previous:** Relating images **Contents**

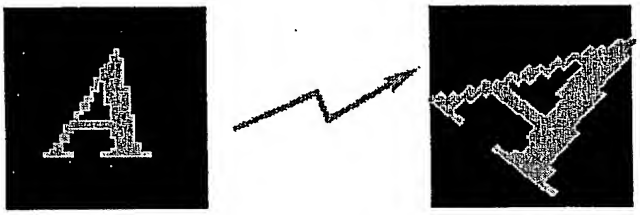
*Marc Pollefeys 2000-07-12*

EXHIBIT 2156.004

# APPENDIX B

4004936 4004936.01

# Affine Transformation



**Common Names:** Affine Transformation

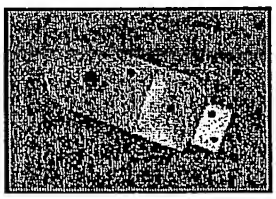
## Brief Description

In many imaging systems, detected images are subject to geometric distortion introduced by perspective irregularities wherein the position of the camera(s) with respect to the scene alters the apparent dimensions of the scene geometry. Applying an affine transformation to a uniformly distorted image can correct for a range of perspective distortions by transforming the measurements from the ideal coordinates to those actually used. (For example, this is useful in satellite imaging where geometrically correct ground maps are desired.)

An affine transformation is an important class of linear 2-D geometric transformations which maps variables (*e.g.* pixel intensity values located at position $(x_1, y_1)$ in an input image) into new variables (*e.g.* $(x_2, y_2)$ in an output image) by applying a linear combination of translation, rotation, scaling and/or shearing (*i.e.* non-uniform scaling in some directions) operations.

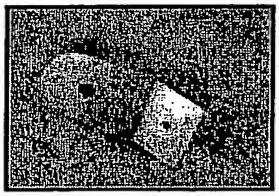
## How It Works

In order to introduce the utility of the affine transformation, consider the image



wherein a machine part is shown lying in a fronto-parallel plane. The circular hole of the part is imaged

as a circle, and the parallelism and perpendicularity of lines in the real world are preserved in the image plane. We might construct a model of this part using these primitives; however, such a description would be of little use in identifying the part from



Here the circle is imaged as an ellipse, and orthogonal world lines are not imaged as orthogonal lines.

This problem of perspective can be overcome if we construct a shape description which is *invariant* to perspective projection. Many interesting tasks within model based computer vision can be accomplished without recourse to Euclidean shape descriptions (*i.e.* those requiring absolute distances, angles and areas) and, instead, employ descriptions involving *relative* measurements (*i.e.* those which depend only upon the configuration's intrinsic geometric relations). These relative measurements can be determined directly from images. Figure 1 shows a hierarchy of planar transformations which are important to computer vision.

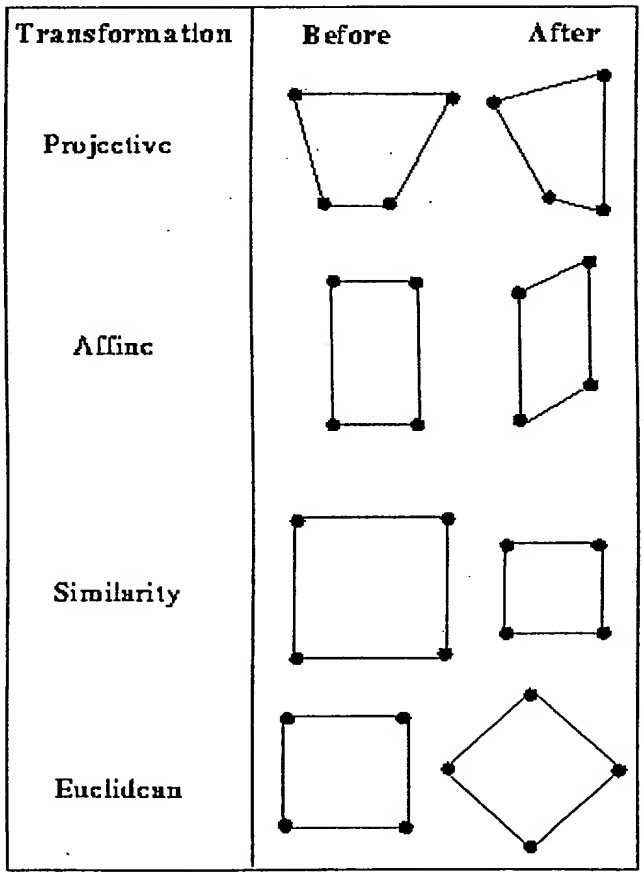
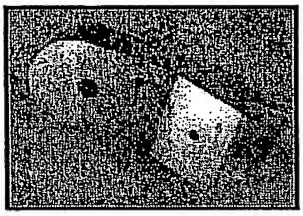



**Figure 1** Hierarchy of plane to plane transformation from Euclidean (where only rotations

and translations are allowed) to Projective (where a square can be transformed into any more general quadrilateral where no 3 points are collinear). Note that transformations lower in the table inherit the invariants of those above, but because they possess their own groups of definitive axioms as well, the converse is not true.

The transformation of the part face shown in the example image above is approximated by a planar *affine* transformation. (Compare this with the image



where the distance to the part is not large compared with its depth and, therefore, parallel object lines begin to converge. Because the scaling varies with depth in this way, a description to the level of *projective* transformation is required.) An affine transformation is equivalent to the composed effects of translation, rotation, isotropic scaling and shear.

The general affine transformation is commonly written in homogeneous coordinates as shown below:

$$\begin{vmatrix} x_2 \\ y_2 \end{vmatrix} = A \times \begin{vmatrix} x_1 \\ y_1 \end{vmatrix} + B$$

By defining only the $B$ matrix, this transformation can carry out pure translation:

$$A = \begin{vmatrix} 1 & 0 \\ 0 & 1 \end{vmatrix}, B = \begin{vmatrix} b_1 \\ b_2 \end{vmatrix}$$

Pure rotation uses the $A$ matrix and is defined as (for positive angles being clockwise rotations):

$$A = \begin{vmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{vmatrix}, B = \begin{vmatrix} 0 \\ 0 \end{vmatrix}$$

Here, we are working in image coordinates, so the y axis goes downward. Rotation formula can be defined for when the y axis goes upward.

Similarly, pure scaling is:

$$A = \begin{vmatrix} a_{11} & 0 \\ 0 & a_{22} \end{vmatrix}, B = \begin{vmatrix} 0 \\ 0 \end{vmatrix}$$

(Note that several different affine transformations are often combined to produce a resultant transformation. The order in which the transformations occur is significant since a translation followed

by a rotation is not necessarily equivalent to the converse.)

Since the general affine transformation is defined by 6 constants, it is possible to define this transformation by specifying the new output image locations $(x_2, y_2)$ of any three input image coordinate $(x_1, y_1)$ pairs. (In practice, many more points are measured and a least squares method is used to find the best fitting transform.)

## Guidelines for Use

Most implementations of the affine operator allow the user to define a transformation by specifying to where 3 (or less) coordinate pairs from the input image $(x_1, y_1)$ re-map in the output image $(x_2, y_2)$. (It is often the case, as with the implementation used here, that the user is restricted to re-mapping *corner coordinates* of the input image to *arbitrary new coordinates* in the output image.) Once the transformation has been defined in this way, the re-mapping proceeds by calculating, for each output pixel location $(x_2, y_2)$, the corresponding input coordinates $(x_1, y_1)$. If that input point is outside of the image, then the output pixel is set to the background value. Otherwise, the value of (i) the input pixel itself, (ii) the neighbor nearest to the desired pixel position, or (iii) a bilinear interpolation of the neighboring four pixels is used.

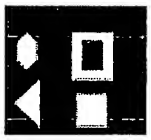
We will illustrate the operation of the affine transformation by applying a series of special-case transformations (*e.g.* pure translation, pure rotation and pure scaling) and then some more general transformations involving combinations of these.

Starting with the 256×256 binary artificial image

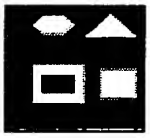
we can apply a translation using the affine operator in order to obtain the image

In order to perform this pure translation, we define a transformation by re-mapping a single point (*e.g.* the input image lower-left corner $(0,0)$ ) to a new position at $(64,64)$.

A pure rotation requires re-mapping the position of two corners to new positions. If we specify that the lower-left corner moves to $(256,0)$ and the lower-right corner moves to $(256,256)$, we obtain



Similarly, reflection can be achieved by swapping the coordinates of two opposite corners, as shown in



Scaling can also be applied by re-mapping just two corners. For example, we can send the lower-left corner to $(64, 64)$, while pinning the upper-right corner down at $(256, 256)$, and thereby uniformly shrink the size of the image subject by a quarter, as shown in

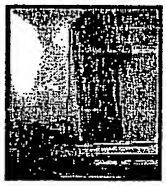
Note that here we have also translated the image. Re-mapping any 2 points can introduce a combination of translation, rotation and scaling.

A general affine transformation is specified by re-mapping 3 points. If we re-map the input image so as to move the lower-left corner up to $(64, 64)$ along the 45 degree oblique axis, move the upper-right corner down by the same amount along this axis, and pin the lower-right corner in place, we obtain an image which shows some shearing effects

Notice how parallel lines remain parallel, but perpendicular corners are distorted.

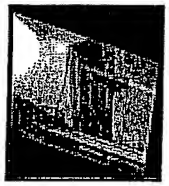
Affine transformations are most commonly applied in the case where we have a detected image which has undergone some type of distortion. The geometrically correct version of the input image can be obtained from the affine transformation by re-sampling the input image such that the information (or intensity) at each point $(x_1, y_1)$ is mapped to the correct position $(x_2, y_2)$ in a corresponding output image.

One of the more interesting applications of this technique is in remote sensing. However, because most images are transformed before they are made available to the image processing community, we will demonstrate the affine transformation with the terrestrial image



which is a contrast-stretched (cutoff fraction = 0.9) version of

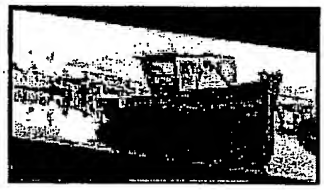
We might want to transform this image so as to map the door frame back into a rectangle. We can do this by defining a transformation based on a re-mapping of the (i) upper-right corner to a position 30% lower along the $y$-axis, (ii) the lower-right corner to a position 10% lower along the $x$-axis, and (iii) pinning down the upper-left corner. The result is shown in



Notice that we have defined a transformation which works well for objects at the depth of the door frame, but nearby objects have been distorted because the affine plane transformation cannot account for distortions at widely varying depths.

It is common for imagery to contain a number of perspective distortions. For example, the original image

shows both affine and projective type distortions due to the proximity of the camera with respect to the subject. After affine transformation, we obtain



Notice that the front face of the captain's house now has truly perpendicular angles where the vertical and horizontal members meet. However, the far background features have been distorted in the process

and, furthermore, it was not possible to correct for the perspective distortion which makes the bow appear much larger than the hull,

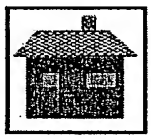
## Interactive Experimentation

You can interactively experiment with this operator by clicking here.

## Exercises

1. It is not always possible to accurately represent the distortion in an image using an affine transformation. In what sorts of imaging scenarios would you expect to find non-linearities in a scanning process and/or differences in along-scans vs across-scans?

2. Apply an affine transformation to the image



**a)** Experiment with different combinations of basic translation, rotation and scaling and then apply a transform which combines several of these operations. **b)** Rotate a translated version of the image and compare your result with the result of translating a rotated version of the image.

## References

**A. Jain** *Fundamentals of Digital Image Processing*, Prentice-Hall, 1986, p 321.

**B. Horn** *Robot Vision*, MIT Press, 1986, pp 314 - 315.

**D. Marr** *Vision*, Freeman, 1982, p 185.

**A. Zisserman** *Notes on Geometric and Invariance in Vision*, British Machine Vision Association and Society for Pattern Recognition, 1992, Chap. 2.

## Local Information

Specific information about this operator may be found here.

More general advice about the local HIPR installation is available in the *Local Information* introductory section.

©2000 R. Fisher, S. Perkins, A. Walker and E. Wolfart.


TOEOT 'SEESOOT

# APPENDIX C

10/15/2004 09:26:00 PM

# Affine transformation matrices

## Name

Affine transformation matrices -- Instantiate and manipulate Affine transforms.

## Synopsis

```
void        art_affine_point            (ArtPoint *dst,
                                         const ArtPoint *src,
                                         const double affine[6]);
void        art_affine_invert           (double dst_affine[6],
                                         const double src_affine[6]);
void        art_affine_flip             (double dst_affine[6],
                                         const double src_affine[6],
                                         int horz,
                                         int vert);
void        art_affine_to_string        (char str[128],
                                         const double src[6]);
void        art_affine_multiply         (double dst[6],
                                         const double src1[6],
                                         const double src2[6]);
void        art_affine_identity         (double dst[6]);
void        art_affine_scale            (double dst[6],
                                         double sx,
                                         double sy);
void        art_affine_rotate           (double dst[6],
                                         double theta);
void        art_affine_shear            (double dst[6],
                                         double theta);
void        art_affine_translate        (double dst[6],
                                         double tx,
                                         double ty);
double      art_affine_expansion        (const double src[6]);
int         art_affine_rectilinear      (const double src[6]);
int         art_affine_equal            (double matrix1[6],
                                         double matrix2[6]);
```

## Description

These API functions allow you to instantiate and manipulate Affine transforms. An Affine transformation is a geometrical transformation which is known to preserve the parallelism of lines but not lengths and angles.

**Figure 1. Affine transforms**

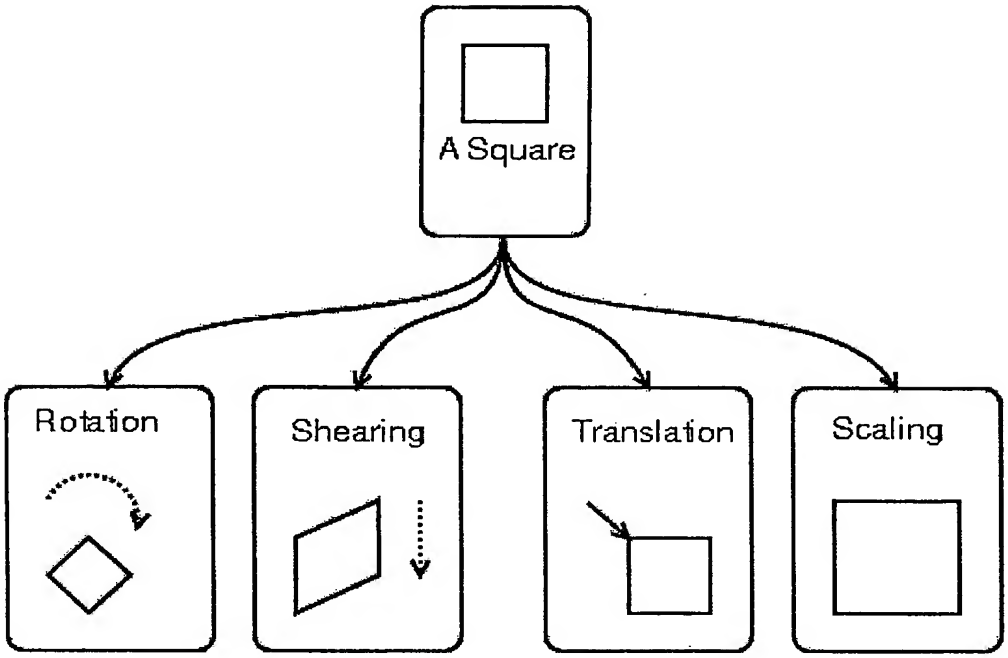



EXHIBIT 5220204

Affine transforms are usually represented using Homogeneous coordinates: given a point (x,y) in the traditional plane, its canonical Homogenous coordinate is (x,y,1). The Affine transforms are represented in Homogeneous coordinates because the transformation of point A by any Affine transformation can be expressed by the multiplication of a 3x3 Matrix and a 3x1 Point vector.

The above property is not trivial. For example, a translation in normal cartesian space results in the addition of a Point vector to the Point vector to transform while in Homogeneous space, the same translation transformation results in a matrix/vector multiplication. To convince you that I am telling you the truth, the diagram below summarizes the different kinds of Affine matrixes used and shows the scaling of point (x,y) by multiplication with a scaling matrix in homogeneous space.

**Figure 2. Affine transform Matrixes in Homogenous space**

Rotation matrix

| | | |
|---|---|---|
| cos o | -sin o | 0 |
| sin o | cos o | 0 |
| 0 | 0 | 1 |

Shear matrix

| | | |
|---|---|---|
| 1 | a | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 1 |

Scaling matrix

| | | |
|---|---|---|
| Sx | 0 | 0 |
| 0 | Sy | 0 |
| 0 | 0 | 1 |

Translation matrix

| | | |
|---|---|---|
| 1 | 0 | dx |
| 0 | 1 | dy |
| 0 | 0 | 1 |

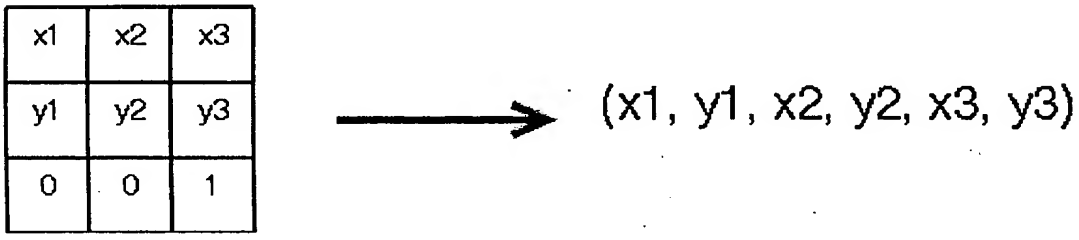
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

with $x' = Sx * x$
and $y' = Sy * y$

To compose two Affine transforms, all you need to do is to multiply their matrices to get the matrix representing the resulting Affine transform. Given Affines A and B represented the matrices MA and MB, the Affine C = AoB is represented by the matrix MC = MA x MB.

Hopefully, all the above gory details (which are unfortunatly necessary to understand the API) are more or less hidden by the API. LibArt thus represents an Affine transform by an array of six doubles:

**Figure 3. An Affine as seen by LibArt**




Those arrays of six doubles can be easily generated with the art_affine_shear, art_affine_scale, art_affine_rotate, art_affine_translate and art_affine_identity functions which generate the affines corresponding to the given transformations. It is possible to composite Affine transformation's matrices with art_affine_multiply and to invert an Affine transformation: art_affine_invert. Finally, I to apply an Affine transformation to a point, you can use art_affine_point

Affine transformations are reused a little everywhere in LibArt: it is possible to apply them to Vector Paths and pixel buffers. art_vpath_affine_transform, art_bpath_affine_transform and art_rgb_affine are such examples.

# Details

## art_affine_point ()

```
void        art_affine_point                (ArtPoint *dst,
                                             const ArtPoint *src,
                                             const double affine[6]);
```

*dst* : Where the result point is stored.

*src* : The original point.

*affine* : The affine transformation.

---

## art_affine_invert ()

```
void        art_affine_invert               (double dst_affine[6],
                                             const double src_affine[6]);
```

All non-degenerate affine transforms are invertible. If the original affine is degenerate or nearly so,

expect numerical instability and very likely core dumps on Alpha and other fp-picky architectures. Otherwise, *dst* multiplied with *src*, or *src* multiplied with *dst* will be (to within roundoff error) the identity affine.

*dst_affine* : Where the resulting affine is stored.

*src_affine* : The original affine transformation.

---

## art_affine_flip ()

```
void        art_affine_flip                 (double dst_affine[6],
                                             const double src_affine[6],
                                             int horz,
                                             int vert);
```

Flips the affine transform. FALSE for both *horiz* and *vert* implements a simple copy operation. TRUE for both *horiz* and *vert* is a 180 degree rotation. It is ok for *src_affine* and *dst_affine* to be equal pointers.

*dst_affine* : Where the resulting affine is stored.

*src_affine* : The original affine transformation.

*horz* : Whether or not to flip horizontally.

*vert* : Whether or not to flip horizontally.

---

## art_affine_to_string ()

```
void        art_affine_to_string            (char str[128],
                                             const double src[6]);
```

Converts an affine transform into a bit of PostScript code that implements the transform. Special cases of scaling, rotation, and translation are detected, and the corresponding PostScript operators used (this greatly aids understanding the output generated). The identity transform is mapped to the null string.

*str* : Where to store the resulting string.

*src* : The affine transform.

---

## art_affine_multiply ()

```
void        art_affine_multiply             (double dst[6],
```

```
                                const double src1[6],
                                const double src2[6]);
```

Multiplies two affine transforms together, i.e. the resulting *dst* is equivalent to doing first *src1* then *src2*. Note that the PostScript concat operator multiplies on the left, i.e. "M concat" is equivalent to "CTM = multiply (M, CTM)";

It is safe to call this function with *dst* equal to *src1* or *src2*.

*dst* : Where to store the result.

*src1* : The first affine transform to multiply.

*src2* : The second affine transform to multiply.

---

## art_affine_identity ()

```
void        art_affine_identity             (double dst[6]);
```

Sets up an identity matrix.

*dst* : Where to store the resulting affine transform.

---

## art_affine_scale ()

```
void        art_affine_scale                (double dst[6],
                                             double sx,
                                             double sy);
```

Sets up a scaling matrix.

*dst* : Where to store the resulting affine transform.

*sx* : X scale factor.

*sy* : Y scale factor.

---

## art_affine_rotate ()

```
void        art_affine_rotate               (double dst[6],
                                             double theta);
```

Sets up a rotation matrix. In the standard libart coordinate system, in which increasing y moves downward, this is a counterclockwise rotation. In the standard PostScript coordinate system, which is reversed in the y direction, it is a clockwise rotation.

*dst* : Where to store the resulting affine transform.

*theta* : Rotation angle in degrees.

---

## art_affine_shear ()

```
void        art_affine_shear                (double dst[6],
                                             double theta);
```

Sets up a shearing matrix. In the standard libart coordinate system and a small value for theta, || becomes \\. Horizontal lines remain unchanged.

*dst* : Where to store the resulting affine transform.

*theta* : Shear angle in degrees.

---

## art_affine_translate ()

```
void        art_affine_translate            (double dst[6],
                                             double tx,
                                             double ty);
```

Sets up a translation matrix.

*dst* : Where to store the resulting affine transform.

*tx* : X translation amount.

*ty* : Y translation amount.

---

## art_affine_expansion ()

```
double      art_affine_expansion            (const double src[6]);
```

Finds the expansion factor, i.e. the square root of the factor by which the affine transform affects area. In an affine transform composed of scaling, rotation, shearing, and translation, returns the amount of scaling.

*src* : The affine transformation.

*Returns* : the expansion factor.

---

## art_affine_rectilinear ()

```
int         art_affine_rectilinear          (const double src[6]);
```

Determines whether *src* is rectilinear, i.e. grid-aligned rectangles are transformed to other grid-aligned rectangles. The implementation has epsilon-tolerance for roundoff errors.

*src* : The original affine transformation.

*Returns* : TRUE if *src* is rectilinear.

---

## art_affine_equal ()

```
int         art_affine_equal                (double matrix1[6],
                                             double matrix2[6]);
```

Determines whether *matrix1* and *matrix2* are equal, with epsilon-tolerance for roundoff errors.

*matrix1* : An affine transformation.

*matrix2* : Another affine transformation.

*Returns* : TRUE if *matrix1* and *matrix2* are equal.



FOTEBOT 6220007

# APPENDIX D

4000326 1004

efunda
engineering fundamentals

Least Square

About Us Trade Show Career News Chat InfoStore SpecSearch A

Statistics
Set Theory
Permutations & Combinations
Probability Theory
Distributions
Reliability
Least Squares
Line
Parabola
Polynomials
Multiple Regression
Resources
Bibliography
Related Suppliers
MathWorks
MCR Associates
Densitron
Eng. Software
Wolfram
more...
Suggested Reading
Login

Copyright © 2001 eFunda

from R&D to Production
Work with the Best
MATHEMATICA IN ENGINEERING

Home Membership Palm Store Forum Search Member What's New
Search All for
Materials Design Center Processes Units & Constants Formulas Mi
brought to you by WOLFRAM RE

## Curve Fitting, Regression

Field data is often accompanied by noise. Even though all control parameters (independent variables) remain constant, the resultant outcomes (dependen vary. A process of quantitatively estimating the trend of the outcomes, also **regression** or **curve fitting**, therefore becomes necessary.

The curve fitting process fits equations of approximating curves to the raw fi Nevertheless, for a given set of data, the fitting curves of a given type are g *NOT unique*. Thus, a curve with a minimal deviation from all data points is d *best-fitting curve* can be obtained by **the method of least squares**.

## The Method of Least Squares

The method of least squares assumes that the best-fit curve of a given type curve that has the minimal sum of the deviations squared (*least square erro* given set of data.

Suppose that the data points are $(x_1,y_1)$, $(x_2,y_2)$, ..., $(x_n,y_n)$ where $x$ is independent variable and $y$ is the dependent variable. The fitting curve $f(x$ deviation (error) $d$ from each data point, i.e., $d_1 = y_1 - f(x_1)$, $d_2 = y_2 - f$ $d_n = y_n - f(x_n)$. According to the method of least squares, the best fitting the property that:

$$\Pi = {d_1}^2 + {d_2}^2 + \ldots + {d_n}^2 = \sum_{i=1}^{n} {d_i}^2 = \sum_{i=1}^{n} [y_i - f(x_i)]^2 = \text{a minimu}$$

## Polynomials Least-Squares Fitting

Polynomials are one of the most commonly used types of curves in regressio applications of the method of least squares curve fitting using polynomials a discussed as follows. To obtain further information on a particular curve fittin

click on the link at the end of each item. Or try the calculator on the right

**The Least-Squares Line**: The least-squares line method uses a *straight lin* $y=a+bx$ to approximate the given set of data, $(x_1,y_1)$, $(x_2,y_2)$, ..., $(x_n$ where $n\geq 2$. See complete derivation.

**The Least-Squares Parabola**: The least-squares parabola method uses a *s degree curve* $y=a+bx+cx^2$ to approximate the given set of data, $(x_1,y_1$ ..., $(x_n,y_n)$, where $n\geq 3$. See complete derivation.

**The Least-Squares $m^{th}$ Degree Polynomials**: The least-squares $m^{th}$ deg Polynomials method uses $m^{th}$ *degree polynomials* $y=a_0+a_1x+a_2x^2+\ldots$ approximate the given set of data, $(x_1,y_1)$, $(x_2,y_2)$, ..., $(x_n,y_n)$, where $n$ See complete derivation.

**Multiple Regression Least-Squares**: Multiple regression estimates the ou which may be affected by more than one control parameter or there may be one control parameter being changed at the same time, e.g., $z=a+bx+c$ complete derivation.


Introduction to Numerical Analysis 2nd ed., *by Hildebrand, F.B.*


Elementary Numerical Analysis 2nd ed., *by Atkinson, K.E.*


Advanced Engi Mathematics 8t *Kreyszig, E.*